Contents

1 Answers

1.1 First 5 rows of each table

host_id	host_name	host_identity_verified
130349612	Dimitrios	unconfirmed
130593431	Emmanouela	verified
131602089	Audrey	unconfirmed
132238305	Karmesha	verified
134952296	Amanda	verified

Figure 1: first 5 rows of Hosts

id	host_id	host_name	name	room_type	minimum_nights
1001254	80014485718	Madaline	Clean & quiet apt home by the park	Private room	10
1002102	52335172823	Jenna	Skylit Midtown Castle	Entire home/apt	13
1002403	78829239556	Elise	THE VILLAGE OF HARLEM....NEW YORK !	Private room	3
1002755	85098326012	Garry	blank	Entire home/apt	13
1003689	92037596077	Lyndon	Entire Apt: Spacious Studio/Loft by central park	Entire home/apt	10

Figure 2: first 5 rows of Listings

id	host_id	host_name	last_review	number_of_reviews	review_rate_number	reviews_per_month
1001254	80014485718	Madaline	2021-10-19	9	4	0.21
1002102	52335172823	Jenna	2022-05-21	45	4	0.38
1002403	78829239556	Elise	2019-06-14	0	5	0.79
1002755	85098326012	Garry	2019-07-05	270	4	4.64
1003689	92037596077	Lyndon	2018-11-19	9	3	0.1

Figure 3: first 5 rows of Reviews

id	lat	long	neighbourhood	neighbourhood_group
1001254	40.64749	-73.97237	Kensington	Brooklyn
1002102	40.75362	-73.98377	Midtown	Manhattan
1002403	40.80902	-73.9419	Harlem	Manhattan
1002755	40.68514	-73.95976	Clinton Hill	Brooklyn
1003689	40.79851	-73.94399	East Harlem	Manhattan

Figure 4: first 5 rows of Locations

id	price	service_fee	cancellation_policy
1001254	966	193	Strict
1002102	142	28	Moderate
1002403	620	124	Flexible
1002755	368	74	Moderate
1003689	204	41	Moderate

Figure 5: first 5 rows of Pricing

1.2 ER model

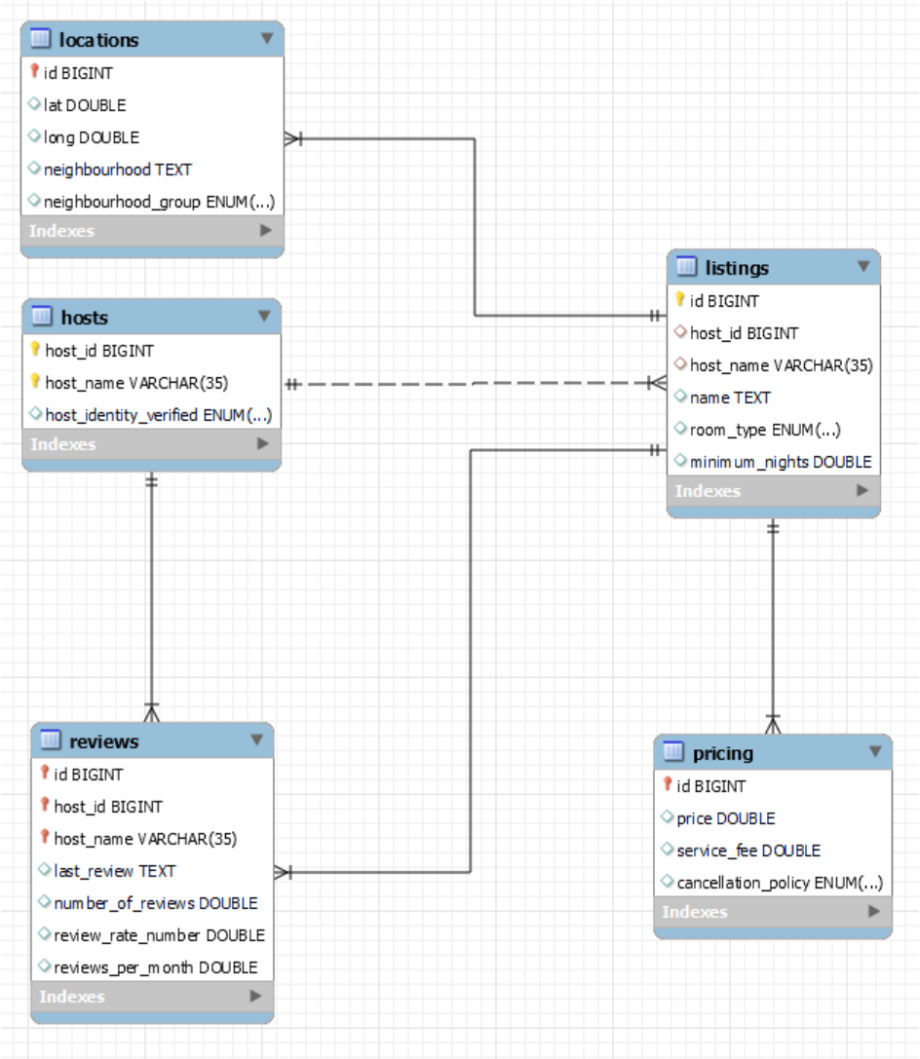

Figure 6: ER-model

2 Specific Steps

2.1 Import Data in mysql

Listing 1: import data in mysql

```
import pandas as pd
from sqlalchemy import create_engine

# read CSV
df = pd.read_csv("C:\\Program␣
    Files\\PyCharm\\manage_big_data\\airbnb_nyc_clean.csv")

# create mysql engine
engine =
    create_engine('mysql+mysqlconnector://root:LnastonChen123456@localhost:3306/bigdata',
    echo=False)

# load data to mysql
df.to_sql(name='airbnb_nyc', con=engine, if_exists='replace',
    index=False)
```

2.2 Explore the Data

2.2.1 create the summary table

create one summary table, to watch the raw column numer,unique column number,Null column number,Max Value,Min Value and Average Value

Listing 2: create the summary table

```
use bigdata;
CREATE TABLE summary_airbnb_nyc_7 (
       Column_name text,
    TotalRowCount INT,
    TotalCount INT,
    NullCount INT,
    UniqueCount INT,
    MaxColumnValue int,
    MinColumnValue int,
    AvgValue int
);
```

2.2.2 populate summary table with data

to some no-numerical column, give Null

Listing 3: populate summary table with data

```
use bigdata;
INSERT INTO summary_airbnb_nyc_7
SELECT
    'availability_365' ,COUNT(*) ,COUNT(availability_365)
        ,COUNT(*) - COUNT(availability_365) ,COUNT(DISTINCT
        availability_365) ,MAX(availability_365)
        ,MIN(availability_365) ,AVG(CASE WHEN availability_365 IS
        NOT NULL THEN availability_365 ELSE NULL END)
FROM airbnb_nyc
UNION ALL
SELECT
    'calculated_host_listings_count', COUNT(*),
        COUNT(calculated_host_listings_count), COUNT(*) -
        COUNT(calculated_host_listings_count), COUNT(DISTINCT
        calculated_host_listings_count),
        MAX(calculated_host_listings_count),
        MIN(calculated_host_listings_count), AVG(CASE WHEN
        calculated_host_listings_count IS NOT NULL THEN
        calculated_host_listings_count ELSE NULL END)
FROM airbnb_nyc
UNION ALL
SELECT
    'cancellation_policy', COUNT(*), COUNT(cancellation_policy),
        COUNT(*) - COUNT(cancellation_policy), COUNT(DISTINCT
        cancellation_policy), NULL, NULL, NULL
FROM airbnb_nyc
UNION ALL
SELECT
    'construction_year', COUNT(*), COUNT(construction_year),
        COUNT(*) - COUNT(construction_year), COUNT(DISTINCT
        construction_year), MAX(construction_year),
        MIN(construction_year), AVG(CASE WHEN construction_year
        IS NOT NULL THEN construction_year ELSE NULL END)
FROM airbnb_nyc
UNION ALL
SELECT
    'host_id', COUNT(*), COUNT(host_id), COUNT(*) -
        COUNT(host_id), COUNT(DISTINCT host_id), NULL, NULL, NULL
FROM airbnb_nyc
UNION ALL
SELECT
```

```
    'host_identity_verified', COUNT(*),
        COUNT(host_identity_verified), COUNT(*) -
        COUNT(host_identity_verified), COUNT(DISTINCT
        host_identity_verified), NULL, NULL, NULL
FROM airbnb_nyc
UNION ALL
SELECT
    'host_name', COUNT(*), COUNT(host_name), COUNT(*) -
        COUNT(host_name), COUNT(DISTINCT host_name), NULL, NULL,
        NULL
FROM airbnb_nyc
UNION ALL
SELECT
    'house_rules', COUNT(*), COUNT(house_rules), COUNT(*) -
        COUNT(house_rules), COUNT(DISTINCT house_rules), NULL,
        NULL, NULL
FROM airbnb_nyc
UNION ALL
SELECT
    'id', COUNT(*), COUNT(id), COUNT(*) - COUNT(id),
        COUNT(DISTINCT id), NULL, NULL, NULL
FROM airbnb_nyc
UNION ALL
SELECT
    'instant_bookable',COUNT(*), COUNT(instant_bookable),
        COUNT(*) - COUNT(instant_bookable), COUNT(DISTINCT
        instant_bookable), NULL, NULL, NULL
FROM airbnb_nyc
UNION ALL
SELECT
    'last_review', COUNT(*), COUNT(last_review), COUNT(*) -
        COUNT(last_review),COUNT(DISTINCT last_review), NULL,
        NULL, NULL
FROM airbnb_nyc
UNION ALL
SELECT
    'lat', COUNT(*), COUNT(lat), COUNT(*) - COUNT(lat),
        COUNT(DISTINCT lat), MAX(lat), MIN(lat), AVG(CASE WHEN
        lat IS NOT NULL THEN lat ELSE NULL END)
FROM airbnb_nyc
UNION ALL
SELECT
    'long', COUNT(*), COUNT(`long`), COUNT(*) - COUNT(`long`),
        COUNT(DISTINCT `long`), MAX(`long`), MIN(`long`),
        AVG(CASE WHEN `long` IS NOT NULL THEN `long` ELSE NULL
        END)
```

```
FROM airbnb_nyc
UNION ALL
SELECT
    'minimum_nights', COUNT(*), COUNT(minimum_nights), COUNT(*) -
        COUNT(minimum_nights), COUNT(DISTINCT minimum_nights),
        MAX(minimum_nights), MIN(minimum_nights), AVG(CASE WHEN
        minimum_nights IS NOT NULL THEN minimum_nights ELSE NULL
        END)
FROM airbnb_nyc
UNION ALL
SELECT
    'name', COUNT(*), COUNT(name), COUNT(*) - COUNT(name),
        COUNT(DISTINCT name), NULL, NULL, NULL
FROM airbnb_nyc
UNION ALL
SELECT
    'neighbourhood', COUNT(*), COUNT(neighbourhood), COUNT(*) -
        COUNT(neighbourhood), COUNT(DISTINCT neighbourhood),
        NULL, NULL, NULL
FROM airbnb_nyc
UNION ALL
SELECT
    'neighbourhood_group', COUNT(*), COUNT(neighbourhood_group),
        COUNT(*) - COUNT(neighbourhood_group), COUNT(DISTINCT
        neighbourhood_group), NULL, NULL, NULL
FROM airbnb_nyc
UNION ALL
SELECT
    'number_of_reviews', COUNT(*), COUNT(number_of_reviews),
        COUNT(*) - COUNT(number_of_reviews), COUNT(DISTINCT
        number_of_reviews), MAX(number_of_reviews),
        MIN(number_of_reviews), AVG(CASE WHEN number_of_reviews
        IS NOT NULL THEN number_of_reviews ELSE NULL END)
FROM airbnb_nyc
UNION ALL
SELECT
    'price', COUNT(*), COUNT(price), COUNT(*) - COUNT(price),
        COUNT(DISTINCT price), MAX(price), MIN(price), AVG(CASE
        WHEN price IS NOT NULL THEN price ELSE NULL END)
FROM airbnb_nyc
UNION ALL
SELECT
    'review_rate_number', COUNT(*), COUNT(review_rate_number),
        COUNT(*) - COUNT(review_rate_number), COUNT(DISTINCT
        review_rate_number), MAX(review_rate_number),
        MIN(review_rate_number), AVG(CASE WHEN review_rate_number
```

```
        IS NOT NULL THEN review_rate_number ELSE NULL END)
FROM airbnb_nyc
UNION ALL
SELECT
    'reviews_per_month', COUNT(*), COUNT(reviews_per_month),
        COUNT(*) - COUNT(reviews_per_month), COUNT(DISTINCT
        reviews_per_month), MAX(reviews_per_month),
        MIN(reviews_per_month), AVG(CASE WHEN reviews_per_month
        IS NOT NULL THEN reviews_per_month ELSE NULL END)
FROM airbnb_nyc
UNION ALL
SELECT
    'room_type', COUNT(*), COUNT(room_type), COUNT(*) -
        COUNT(room_type), COUNT(DISTINCT room_type), NULL, NULL,
        NULL
FROM airbnb_nyc
UNION ALL
SELECT
    'service_fee', COUNT(*), COUNT(service_fee), COUNT(*) -
        COUNT(service_fee), COUNT(DISTINCT service_fee),
        MAX(service_fee), MIN(service_fee), AVG(CASE WHEN
        service_fee IS NOT NULL THEN service_fee ELSE NULL END)
FROM airbnb_nyc;
```

2.3 Explore data results

Column_name	TotalRowCount	TotalCount	NullCount	UniqueCount	MaxColumnValue	MinColumnValue	AvgValue
construction_year	69305	69305	0	20	2022	2003	2012
price	69305	69305	0	1152	1200	50	625
availability_365	69305	69305	0	437	426	-10	153
service_fee	69305	69305	0	232	240	10	125
lat	69305	69305	0	21998	41	40	41
number_of_reviews	69305	69305	0	476	1024	0	28
calculated_host_listings_count	69305	69305	0	78	332	1	9
minimum_nights	69305	69305	0	14	13	0	5
review_rate_number	69305	69305	0	5	5	1	3
reviews_per_month	69305	69305	0	1016	90	0	1
long	69305	69305	0	17780	-74	-74	-74
cancellation_policy	69305	69305	0	3	NULL	NULL	NULL
host_id	69305	69305	0	69304	NULL	NULL	NULL
host_identity_verified	69305	69305	0	2	NULL	NULL	NULL
host_name	69305	69305	0	13069	NULL	NULL	NULL
house_rules	69305	69305	0	1959	NULL	NULL	NULL
id	69305	69305	0	69305	NULL	NULL	NULL
instant_bookable	69305	69305	0	2	NULL	NULL	NULL
last_review	69305	69305	0	2472	NULL	NULL	NULL
name	69305	69305	0	60604	NULL	NULL	NULL
neighbourhood	69305	69305	0	239	NULL	NULL	NULL
neighbourhood_group	69305	69305	0	5	NULL	NULL	NULL
room_type	69305	69305	0	4	NULL	NULL	NULL

Figure 7: explore data results

We discover some important imformation from exploring

1. There are negative values in availability_365 that need to be cleaned or the data source investigated
2. construction_year has a minimum value of 2003 and a maximum value of 2022, which seems like a reasonable range
3. host_id and id are both unique and meet the requirements for a primary ke
4. cancellation_policy and room_type have a very small number of unique values, suitable for enumeration or as a foreign key
5. neighborhood_group has 5 unique values and is suitable for use as a foreign key
6. price and service_fee have very different minimum and maximum values and may require further investigation
7. There are a total of 69,305 pieces of data, using host_id as the primary key then there may be duplicate values, check and find a host_id corresponds to 2 records, so found that you need to use host_id + host_name as the primary key

2.4 Identify Data Elements

below are all the data elements:
availability_365
calculated_host_listings_count
cancellation_policy
construction_year
host_id
host_name
host_identity_verified
house_rules
id (Listing ID)
instant_bookable
last_review
lat
long
minimum_nights
name (Listing Name)
neighbourhood
neighbourhood_group
number_of_reviews
price
review_rate_number
reviews_per_month
room_type
service_fee

2.5 Divide Data Elements

Data elements can be categorized into the following distinct areas or concepts:

1. Host information
2. Listing Information
3. Review information
4. Pricing information

2.6 Identify Tables

Depending on the field or concept, the following tables can be created:

1. Hosts
2. Listings
3. Reviews
4. Pricing

2.7 Identify Keys

1. Hosts: (host_id, host_name)
2. Listings: id
3. Reviews: (id, host_id, host_name)
4. Locations: (id)
5. Pricing: (id)

2.8 Normalization

The data appears to be under 3NF already, as there is no observable redundant data and each field is dependent on the primary key.
In addition to the 6 issues we found when exploring the data, as follows (Normalization is done when populating the 3NF table with data)

1. There are negative values in availability_365 that need to be cleaned or the data source investigated
2. construction_year has a minimum value of 2003 and a maximum value of 2022, which seems like a reasonable range
3. host_id and id are both unique and meet the requirements for a primary ke
4. ....

2.9 Create tables

Listing 4: create 3NF table

```
-- Hosts Table
CREATE TABLE Hosts (
  host_id BIGINT ,
  host_name varchar(35),
  host_identity_verified ENUM('unconfirmed','verified'),
  PRIMARY KEY(host_id, host_name),
  CHECK(host_identity_verified IS NOT NULL)
);

-- Listings Table
CREATE TABLE Listings (
  id BIGINT ,
  host_id BIGINT ,
  host_name varchar(35),
  `name` TEXT,
  room_type ENUM('Private␣room','Entire␣home/apt','Shared␣
      room','Hotel␣room'),
  minimum_nights DOUBLE CHECK (minimum_nights >= 0),
  PRIMARY KEY(id),
  FOREIGN KEY(host_id, host_name) REFERENCES Hosts(host_id,
      host_name)
);

-- Reviews Table
CREATE TABLE Reviews (
  id BIGINT ,
  host_id BIGINT ,
  host_name varchar(35),
  last_review TEXT,
  number_of_reviews DOUBLE CHECK (number_of_reviews >= 0),
  review_rate_number DOUBLE CHECK (review_rate_number BETWEEN 1
      AND 5),
  reviews_per_month DOUBLE CHECK (reviews_per_month >= 0),
  PRIMARY KEY(id, host_id, host_name),
  FOREIGN KEY(id) REFERENCES Listings(id),
  FOREIGN KEY(host_id, host_name) REFERENCES Hosts(host_id,
      host_name)
);

-- Locations Table
CREATE TABLE Locations (
  id BIGINT ,
```

```
  lat DOUBLE ,
  `long` DOUBLE ,
  neighbourhood TEXT,
  neighbourhood_group
      ENUM('Brooklyn','Manhattan','Queens','Staten␣
      Island','Bronx'),
  PRIMARY KEY(id),
  FOREIGN KEY(id) REFERENCES Listings(id)
);

-- Pricing Table
CREATE TABLE Pricing (
  id BIGINT ,
  price DOUBLE CHECK (price >= 0),
  service_fee DOUBLE CHECK (service_fee >= 0),
  cancellation_policy ENUM('Flexible', 'Moderate', 'Strict'),
  PRIMARY KEY(id),
  FOREIGN KEY(id) REFERENCES Listings(id)
);
```

2.10 Develop Index

Create indexes for host_id and host_name on the Hosts table. Create indexes on the Listings and Pricing tables for id.

Listing 5: create Index

```
-- Index Creation
CREATE INDEX idx\_host ON Hosts(host\_id, host\_name);
CREATE INDEX idx\_listing ON Listings(id);
CREATE INDEX idx\_review ON Reviews(id, host\_id, host\_name);
CREATE INDEX idx\_location ON Locations(id);
CREATE INDEX idx\_pricing ON Pricing(id);
```